AI for Satellite Collision Avoidance – Go/No Go Decision-Making

Pavithra Ravi[1], Andrea Zollo[2], and Hauke Fiedler[3]

[1,2,3] German Space Operations Center (GSOC), DLR, Münchener Str. 20, Weßling, Germany
Pavithra.Ravi@dlr.de, Andrea.Zollo@dlr.de, Hauke.Fiedler@dlr.de

Abstract

The proliferation of manmade objects in orbit for commercial, defense, and research purposes has resulted in frequent collision alerts. While much of the conjunction assessment and collision avoidance pipeline is automated, the decision on whether or not to carry out an avoidance maneuver is typically left to satellite operators. This work explores the potential of automating this *go*/*no go* decision using artificial intelligence (AI). A set of high-risk Conjunction Data Messages (CDMs) pertaining to 200 events is provided to analysts who are tasked with manually classifying them with a *go*/*no go* for maneuever. These events have a probability of collision (PoC) close to the critical threshold of 1E-4, treading the line between a typical *go* and *no go* decision. The classification results from the analysts are used as training data for AI models. Feature engineering and hyperparameter optimization is done to ensure the models are well equipped to learn from this data. The models' abilities to identify relevant patterns from the CDMs and correctly classify unseen CDMs into the two classes is assessed. This work discusses the characteristics and processing of the CDM population used, the results from the developed models, and the insights gained on the practicability of applying AI to decision-making in this context.

1 INTRODUCTION

Operational payloads in Low Earth Orbit (LEO) are growing at an exponential rate – in 2019, there were 800 active satellites in LEO, a number which has grown to over 5000 as of 2022 [1]. Current launch traffic patterns, continued fragmentations, and limited Post-Mission Disposal (PMD) rates are resulting in an imminent cascade of critical conjunctions [2]. Nearly 43,000 high Probability of Collision (PoC) encounters with a criticality greater than 1E-6 occur monthly in LEO [3]. Given an additional 58,000 active satellites are to be launched by 2030 [4], high PoC events are likely to become commonplace in a crowded orbital regime, warranting frequent collision avoidance activities. Collision avoidance that relies on satellite operators around the clock to make decisions might not be scalable to account for these trends. Thus, this work explores the applicability of Artificial Intelligence (AI) to a central component of collision avoidance practices – *go*/*no go* decision-making.

Recent efforts to apply AI to risk assessment in collision avoidance have been made, most notably through European Space Agency's (ESA) 2019 Spacecraft Collision Avoidance Challenge, which tasked participants with predicting the final collision risk (PoC) between two objects using Conjunction Data Messages (CDMs) [5]. AI models to predict final risk have also been investigated by [6] and [7] – these are assessed on their ability to correctly classify events as having high or low final risk using a threshold of 1E-6, as was used in ESA's competition. Additionally, deep learning techniques, such as the use of Long Short-Term Memory (LSTM) models, have been tested by [8], [7], and [9] for the use of a sequence of CDMs to predict features of interest (such as PoC and covariance) in future CDMs. Furthermore, multi-class classification approaches to classify a CDM based on its likelihood to change its risk category [10], and the use of Evidence Theory to classify encounter geometries into five classes based on event criticality [11] have been put forth.

This work innovates by developing AI models to classify critical events – high risk events with PoC mostly between 1E-5 to 1E-4 – that would typically involve the assessment of a human operator. AI models are likely not needed for events with much lower PoC (less than 1E-6), or for events with much higher PoC (greater than 1E-3) as these would presumably be straightforward *no go*, and *go* decisions respectively, which can be automated using simple *if*-statements. For an automated decision-making system to truly lessen the burden on operators, developed models have to ultimately make reliable decisions for borderline cases autonomously, or at least serve as a trustworthy recommendation system that can lessen operator workload. Thus, this work investigates the extent to which AI models are able to match critical decisions made by satellite operators at the German Space Operations Center (GSOC).

2 CRITICAL EVENTS GENERATION AND CLASSIFICATION

The dataset from ESA's 2019 Spacecraft Collision Avoidance Challenge is used in this work [5]. Each CDM in the dataset contains 103 features including probability of collision, time to Time of Closest Approach (TCA), miss distance, covariance information, and other parameters which can aid characterizing the criticality of an event [5]. To train AI models in this work, 200 events are classified by GSOC satellite operators to obtain a 'ground truth'.

In addition to quantities in the CDMs, analysts also consider other parameters (such as the *K*-factor, Maximum PoC, as highlighted in Section 2.3) as part of their decision. At GSOC, these are obtained after processing the CDMs using an in-house tool, the Collision Avoidance System (CAS). However, for these outputs to be generated, CDMs have to have complete state vector information of the two objects. Since the Kelvin data was completely anonymized, this information was left out of the dataset. However, three orbital elements (semi-major axis, inclination, and eccentricity) are provided for both the primary and secondary objects along with their relative state vector – these are used to reverse-engineer synthetic states. Section 2.1 outlines the events from the dataset that are used in this work and Section 2.2 presents the method used to synthetically generate state vectors for each CDM.

2.1 Data Cleaning and Pre-Processing

ESA's competition dataset comprises 13,154 events in the training set and 2,167 events in the test set. The events in the test set consist of events corresponding to useful operational scenarios and only contain CDMs released at least two days before the TCA [5]. While the training set CDMs are not capped by time to TCA, only CDMs released at least one day before TCA are considered in this work, to better mimic operational conditions. Given the AI model is to be tested on events that would typically involve an operator, critical events with a PoC > 1E-5 are screened for. 73 of the 2,167 test set events have a PoC > 1E-5 in the last CDM and 112 of the 13,154 training set events have a PoC > 1E-5 in the last CDM that is at least 1 day from TCA. Two additional constants are applied to the shortlisted 1E-5 events: the events need to have at least 3 CDMs to allow for sufficient temporal information, and also contain complete covariance information, which is found to be lacking for the secondary object in certain events. 78 events in the training set and 64 events in the test set meet these criteria, resulting in 142 events. The remaining 58 events to attain the desired 200 events for this project are selected by sorting the remaining test set events according to PoC and selecting the 58 most high-risk events with at least 3 CDMs and complete covariance information. All CDM series are capped at 6 CDMs to lessen the complexity of inputs fed to the AI models – it is assumed sufficient critical temporal information is preserved within the last 6 CDMs to TCA.

2.2 Synthetic State Vector Generation

Input CDMs to CAS need to have complete state vector information. A method to obtain a pair of state vectors for the two objects using their semimajor axes, eccentricities, and inclinations, along with the relative state vector between the objects, is used as shown in Fig. 1:

```
Algorithm State Vector Computation for Primary and Secondary Objects
 1: Parameters:
 2: semimajor axis, eccentricity, inclination of both primary and secondary
    objects $givenPriKeplerian$, $givenSecKeplerian$, relative state vector
    $relState$, Cartesian state function $Keplerian2Cartesian$, Keplerian state
    function $Cartesian2Keplerian$, weighted difference function $weightedDiff$
 3: $RAAN \leftarrow 0$, restricting right ascension of ascending node for simplicity
 4: $minDifference \leftarrow \infty$
 5: $bestArgP \leftarrow 0$, argument of periapsis
 6: $bestTA \leftarrow 0$, true anomaly
 7: for $argP$ from 0 to 360 in increments of 0.5 do
 8:     for $TA$ from 0 to 360 in increments of 0.5 do
 9:         $priKeplerian \leftarrow [givenPriKeplerian, RAAN, bestArgP, bestTA]$
10:         $priCartesian \leftarrow Keplerian2Cartesian(priKeplerian)$
11:         $secCartesian \leftarrow priCartesian + relState$
12:         $secKeplerian \leftarrow Cartesian2Keplerian(secCartesian)$
13:         $[\Delta a, \Delta e, \Delta i] \leftarrow weightedDiff(givenSecKeplerian, secKeplerian)$
14:         $\Delta_{total} \leftarrow \Delta a + \Delta e + \Delta i$
15:         if $\Delta_{total} < minDifference$ then
16:             $minDifference \leftarrow \Delta_{total}$
17:             $bestArgP \leftarrow argP$
18:             $bestTA \leftarrow TA$
19:         end if
20:     end for
21: end for
22: for $argP$ from $(bestArgP - 0.5)$ to $(bestArgP + 0.5)$ in increments of 0.01
    do
23:     for $TA$ from $(bestTA - 0.5)$ to $(bestTA + 0.5)$ in increments of 0.01 do
24:         Repeat steps 9-18 to refine $argP$ and $TA$ values.
25:     end for
26: end for
27: $priKeplerian \leftarrow [givenPriKeplerian, RAAN, bestArgP, bestTA]$
28: $priCartesian \leftarrow Keplerian2Cartesian(priKeplerian)$
29: $secCartesian \leftarrow priCartesian + relState$
```

Fig. 1. Algorithm used to generate synthetic state vectors while preserving relative encounter geometry.

The algorithm fixes one orbital element (Right Ascension of Ascending Node, RAAN), determines the values of the remaining orbital elements for both objects such that the relative state is preserved, and subsequently converts these to cartesian state vectors of the two objects.

A weighted difference is used to assess the closeness of the estimated secondary object's elements relative to the given elements. This accounts for the difference in the scale of semimajor axis, a, in kilometers, compared to eccentricity, e, and inclination, i, in radians. The weighted difference, $\Delta total$, is calculated as,

$$\Delta total = \Delta a + \Delta e + \Delta i \tag{1}$$

Where,

$$\Delta a = |a_{estimated} - a_{given}| \tag{2}$$

$$\Delta e = |e_{estimated} - e_{given}| \times a_{given} \tag{3}$$

$$\Delta i = \frac{|i_{estimated} - i_{given}| \times a_{given}}{\pi} \tag{4}$$

There are two positions along the orbits where the close approach can occur – two pairs of argument of periapsis (ω) and true anomaly (ν) can both be selected as the 'best' options using the algorithm. These are, ω and $2\pi - \omega$, and ν and $2\pi - \nu$, respectively. It is ensured that the same solution is selected for CDMs belonging to the same event, such that the location of the conjunction event remains consistent.

2.3 CDM Processing and Classification by GSOC

At GSOC, the Collision Avoidance System (CAS) results in various outputs for analysts' consideration: a summary text-file with key parameters across multiple CDMs, collision geometry and CDM-history plots showing the evolution of the position and uncertainties of the objects over each CDM update, and plots depicting the relative position and combined covariance projection onto the B-plane [12]. The PoC is recomputed using updated covariances from improved Orbit Determination (OD) processes and to include any planned maneuvers that are not already accounted for. While Hard Body Radius (HBR) information is available in the competition CDMs through the *x_span* feature, alternate HBRs are employed in the calculation of the PoC to artificially increase the number of critical events. Specifically, the HBR for many events is tweaked to result in a final CDM falling around the 1E-4 PoC threshold – the typical *go*/*no go* threshold used by decision-making analysts at GSOC. Additionally, for events that are originally in the test set of the competition dataset (where the events are capped at 2 days to TCA), the time to TCA values is reduced by 0.6 days to better represent the decision-making timelines typically used by analysts.

The outputs from CAS for the 200 events are distributed amongst 5 analysts in the Flight Dynamics team at GSOC, who each classify 40 events as *go* or *no go* for maneuver. 84 of the 200 events are assigned a *go* for maneuver. Important outputs of the processing tool include the covariance scaling factor, *K*, which indicates if the PoC is diluted [22], and the maximum PoC (max. PoC) for the event. If $K < 1$, which implies a greater PoC if uncertainties are reduced, the max. PoC for the event is used by the analysts to make the decision. These classified results from the analysts are considered the target labels which the AI models are subsequently trained to reproduce. Figure 2 shows the distribution of the PoC in the last CDM across the 200 events. "PoC used for decision" incorporates the consideration of the *K*-factor, and it can be observed that multiple events are then perceived as more critical. It is worth noting that not all events with PoC > 1E-4 are assigned a *go*. Large covariance, untrustworthy OD parameters, or jumps in the CDM series, can result in a *no go*. It is of particular interest to ascertain whether AI models are able to match such decisions which are influenced by a myriad of features.

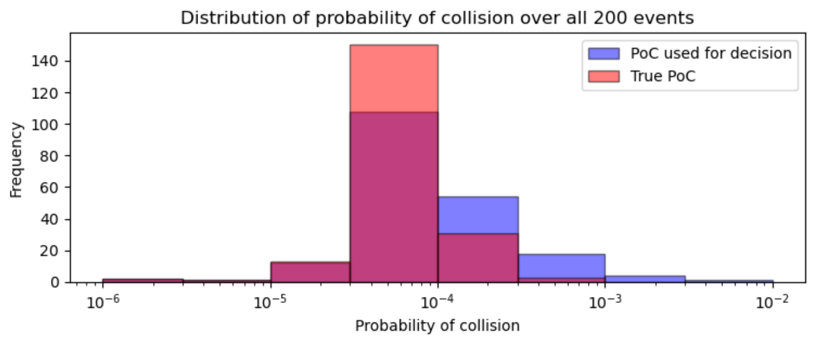


Fig. 2. PoC distribution across events. "PoC used for decision" replaces the true PoC of events with a covariance scaling factor, *K*, < 1, with the Max. PoC for that event.

3 AI FOR TIME SERIES CLASSIFICATION

In this work, the go/no-go decision-making problem is approached as a time series classification (TSC) task. This provides the AI model the same temporal information that analysts rely on when making their decision. Unlike traditional classification algorithms which tend to assume conditional independence of features, TSC algorithms can account for the correlation between data in consecutive time steps [13]. The data in this work is multivariate, owing to different features considered, and includes CDM series with varying lengths (3-6 CDMs). TSC algorithms are generally catered to univariate and fixed-length series, with few algorithms equipped to directly accommodate multivariate varying-length series [14]. This work uses tools from *tslearn* [15], a machine learning library in Python with algorithms for TSC. Additionally, *keras* [16], a deep learning API in Python with *tensorflow* [17] backend is used for implementing an LSTM model.

3.1 Candidate Models

This section describes the AI models investigated in this work.

3.1.1 *K*-Nearest Neighbors with Dynamic Time Warping (KNN-DTW)

KNN-DTW is a variation of the standard KNN method, which relies on a distance measure that can accommodate varying length series, namely Dynamic Time Warping (DTW) [18]. It is shown by [19] to be a strong baseline measure for TSC and is thus employed in this work through *tslearn's KNeighborsTimeSeriesClassifier* with the '*dtw*' metric [15].

3.1.2 *K*-Nearest Neighbors with Symbolic Aggregate Approximation (KNN-SAX)

Symbolic Aggregate Approximation (SAX) represents time series data in the form of symbolic "words" [20]. Each sequence is encapsulated in a symbolic word of length *w* comprising *a* unique alphabets. In order to apply the KNN algorithm with SAX-transformed series, the MINDIST function [20] is used, which returns the minimum distance between the original time series of two symbolic words. For this work, *tslearn's KNeighborsTimeSeriesClassifier* with the '*sax*' metric is used.

3.1.3 Support Vector Machine with Global Alignment Kernel (SVM-GAK)

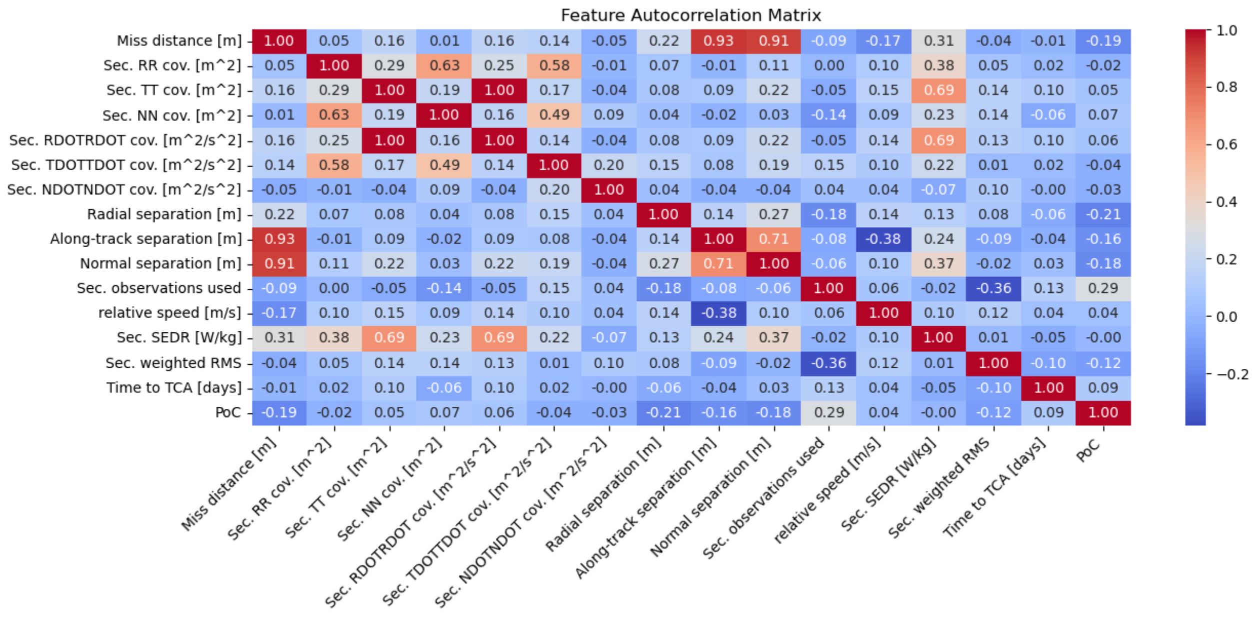
The Support Vector Machine (SVM) algorithm finds an optimal hyperplane that best separates the data points into different classes, while maximizing the margin between the hyperplane and the nearest data points of each class. Reference [21] put forth the Global Alignment Kernel (GAK) for time series data, which quantifies the similarity between two sequences through a summation score of all possible global alignments between them. *Tslearn's TimeSeriesSVC* with the *'gak'* kernel is used in this work.

3.1.4 Long Short-Term Memory (LSTM)

Deep learning techniques, such as the use of Recurrent Neural Networks (RNNs) are also applicable to multivariate time series classification problems due to their ability to handle sequential data. LSTM networks [24] are a specific type of RNNs which possess more complex internal gating mechanisms, allowing them to bypass the vanishing gradient problem which conventional RNNs are prone to. In this work, the LSTM model is built using *keras* and *tensorflow* and consists of a masking layer (to ignore padded values for events with fewer CDMs), a hidden layer with LSTM node units (neurons) to learn trends from the data, a dropout layer for regularization, and lastly a singular output node with sigmoid activation to classify the event.

3.2 Feature Engineering

Parameters that are known to be influential in the analysts' decisions are handpicked as features for the AI models. The PoC feature is the same as the PoC value considered by analyst (true PoC or Max. PoC depending on the *K*-factor as mentioned in Section 2.3). Additionally, features indicative of perturbations on the satellite (namely the satellite energy dissipation rate, SEDR) as well as the quality of the orbit determination process (weighted root-mean-squared error, RMS, and observations used) are included. Only features pertaining to the secondary object, and those accounting for the relative encounter are used, to allow for the model to generalize across missions as validated by [5]. The shortlisted 16 features in the 6-CDM series along with their autocorrelation matrix is presented in Fig. 3. Note that features prefaced with 'Sec.' pertain to the secondary object and 'covariance' is shortened to 'cov.' R, T, and N covariance terms refer to the uncertainties in radial, along-track, and normal directions. Terms with 'DOT', such as in 'RDOT', pertain to velocity terms in the respective RTN frame.



	Miss distance [m]	Sec. RR cov. [m^2]	Sec. TT cov. [m^2]	Sec. NN cov. [m^2]	Sec. RDOTRDOT cov. [m^2/s^2]	Sec. TDOTTDOT cov. [m^2/s^2]	Sec. NDOTNDOT cov. [m^2/s^2]	Radial separation [m]	Along-track separation [m]	Normal separation [m]	Sec. observations used	relative speed [m/s]	Sec. SEDR [W/kg]	Sec. weighted RMS	Time to TCA [days]	PoC
Miss distance [m]	1.00	0.05	0.16	0.01	0.16	0.14	-0.05	0.22	0.93	0.91	-0.09	-0.17	0.31	-0.04	-0.01	-0.19
Sec. RR cov. [m^2]	0.05	1.00	0.29	0.63	0.25	0.58	-0.01	0.07	-0.01	0.11	0.00	0.10	0.38	0.05	0.02	-0.02
Sec. TT cov. [m^2]	0.16	0.29	1.00	0.19	1.00	0.17	-0.04	0.08	0.09	0.22	-0.05	0.15	0.69	0.14	0.10	0.05
Sec. NN cov. [m^2]	0.01	0.63	0.19	1.00	0.16	0.49	0.09	0.04	-0.02	0.03	-0.14	0.09	0.23	0.14	-0.06	0.07
Sec. RDOTRDOT cov. [m^2/s^2]	0.16	0.25	1.00	0.16	1.00	0.14	-0.04	0.08	0.09	0.22	-0.05	0.14	0.69	0.13	0.10	0.06
Sec. TDOTTDOT cov. [m^2/s^2]	0.14	0.58	0.17	0.49	0.14	1.00	0.20	0.15	0.08	0.19	0.15	0.10	0.22	0.01	0.02	-0.04
Sec. NDOTNDOT cov. [m^2/s^2]	-0.05	-0.01	-0.04	0.09	-0.04	0.20	1.00	0.04	-0.04	-0.04	0.04	0.04	-0.07	0.10	-0.00	-0.03
Radial separation [m]	0.22	0.07	0.08	0.04	0.08	0.15	0.04	1.00	0.14	0.27	-0.18	0.14	0.13	0.08	-0.06	-0.21
Along-track separation [m]	0.93	-0.01	0.09	-0.02	0.09	0.08	-0.04	0.14	1.00	0.71	-0.08	-0.38	0.24	-0.09	-0.04	-0.16
Normal separation [m]	0.91	0.11	0.22	0.03	0.22	0.19	-0.04	0.27	0.71	1.00	-0.06	0.10	0.37	-0.02	0.03	-0.18
Sec. observations used	-0.09	0.00	-0.05	-0.14	-0.05	0.15	0.04	-0.18	-0.08	-0.06	1.00	0.06	-0.02	-0.36	0.13	0.29
relative speed [m/s]	-0.17	0.10	0.15	0.09	0.14	0.10	0.04	0.14	-0.38	0.10	0.06	1.00	0.10	0.12	0.04	0.04
Sec. SEDR [W/kg]	0.31	0.38	0.69	0.23	0.69	0.22	-0.07	0.13	0.24	0.37	-0.02	0.10	1.00	0.01	-0.05	-0.00
Sec. weighted RMS	-0.04	0.05	0.14	0.14	0.13	0.01	0.10	0.08	-0.09	-0.02	-0.36	0.12	0.01	1.00	-0.10	-0.12
Time to TCA [days]	-0.01	0.02	0.10	-0.06	0.10	0.02	-0.00	-0.06	-0.04	0.03	0.13	0.04	-0.05	-0.10	1.00	0.09
PoC	-0.19	-0.02	0.05	0.07	0.06	-0.04	-0.03	-0.21	-0.16	-0.18	0.29	0.04	-0.00	-0.12	0.09	1.00

Fig. 3. Autocorrelation matrix showing the interdependence of 16 key features.

Feature engineering in the form of clipping and scaling is performed to account for outliers and to aid the learning processes of the models by constraining the range of values. Clipping refers to the process of assigning an upper and/or lower limit for a feature, and setting all values outside of that range to equal the value of the limit. In this work, all features are assigned an upper limit of 3 standard deviations from the mean except for time to TCA and relative velocity which contain no outliers. Additionally, PoC values are clipped to a minimum value of 1E-6, to prevent uncritical CDMs with near-zero risks from significantly influencing the developed models. Two scaling methods are applied to the clipped data: Z-Normalization and Min-Max. Z-Normalization normalizes the data to have a mean of 0 and a standard deviation of 1. Min-Max scaling linearly scales numbers to fall between a desired range – 0 and 1 are chosen in this work. KNN-DTW, SVM-GAK, and the LSTM methods are tested with both scaling methods. KNN-SAX is only implemented with Z-Normalization to ensure the symbols summarizing segments are on a consistent scale. It is found that different models perform better with different scaling techniques as shown in Section 4.1.

The input feature matrix to the models is set up in the form of a 3D tensor with dimensions (200,6,16), where the data consists of 200 total samples (events), with a maximum of 6 timesteps (CDMs), and 16 features. The classification results from the analysts, or the target labels, are obtained through one-hot encoding – where *go's* are assigned a 1 and *no go's* are assigned a 0.

3.3 Hyperparameter Optimization

In this work, hyperparameter optimization is done using Grid Search, a technique where the model's performance is exhaustively evaluated for all combinations of hyperparameters of interest. Additionally, models are also tested with both Z-normalization and Min-Max scaling. To ensure the results are not biased due to data partitioning, stratified K-fold cross validation with 5 folds is applied along with the grid search. Thus, each combination of hyperparameters is evaluated 5 times on the full dataset of 200 events, where each time a different 1/5 of the dataset (20 %) is in the test set. It is ensured that each test split contains as close to an equal percentage-split of the two classes as possible. The combination of hyperparameters which consistently yields the best F1-score, a metric that accounts for class imbalance [23], is then selected. This process is implemented using *sklearn*'*s GridSearchCV* function in Python. Table 1 summarizes the hyperparameter values selected for the 4 models:

Tab. 1. Hyperparameters tested and selected using a Grid Search with stratified 5-fold cross validation.

Hyperparameter	**Values tested**	**Optimal value**
Neighbors, *k* (KNN-DTW)	1:1:10	6
Weights (KNN-DTW)	Uniform, Distance	Distance (closer neighbors weighted more)
Neighbors, *k* (KNN-SAX)	1:1:10	5
Weights (KNN-SAX)	Uniform, Distance	Uniform (neighbors weighted equally)
Segments, *w* (KNN-SAX)	1:1:3	3
Alphabet size, *a* (KNN-SAX)	5:1:15	7
Penalty parameter, *C* (SVM-GAK)	1:1:10	2
Kernel coefficient, γ (SVM-GAK)	600:100:1200	1000
Neurons (LSTM)	80:20:200	160
Dropout (LSTM)	0.1:0.1:0.7	0.5
Learning rate (LSTM)	0.01:0.01:0.05	0.02
Batch size (LSTM)	2:2:10	2

It is important to highlight that this hyperparameter search is by no means exhaustive. A finer grid search, searching between larger ranges of values using smaller step sizes could potentially yield improved results, but will increase the computation time. For an initial assessment on AI models' abilities for decision-making, the hyperparameters presented in this section are considered to be a decent start towards building robust models.

4 RESULTS

4.1 Model Performance

Results from the best models using the optimized hyperparameters are presented in this section. Table 2 shows the averaged results obtained by running the models over 5 shuffles of the dataset, where a stratified 5-fold cross validation is applied on each shuffle. Thus, each model is tested a total of 25 times to ensure result consistency. The seeds used to shuffle the dataset are constant for each model run such that the models are evaluated on the same arrangement and partition of samples.

Tab. 2. Model performances using optimized hyperparameters averaged over 25 runs including cross validation.

Model (scaling)	Accuracy	Precision	Recall	F1
KNN-DTW (Min-Max)	0.67	0.63	0.56	0.59
KNN-SAX (Z-Norm)	0.66	0.62	0.55	0.58
SVM-GAK (Z-Norm)	0.63	0.55	**0.70**	0.61
LSTM (Z-Norm)	**0.72**	**0.69**	0.65	**0.65**

The metrics to assess the performance of each model include accuracy, precision, recall, and the F1-score [23]. Accuracy refers to the percentage of predicted labels which match the target labels. Because of the class imbalance in this work, a model only guessing *no go* would already attain about 60 % accuracy. Thus, other metrics are more useful to assess the model performance: precision is the percentage of actual *go's* amidst predicted *go's* – a high precision value indicates fewer false positives. Recall is the percentage of predicted *go's* amidst all actual *go's* in the test set – a high recall value implies fewer false negatives. The F1-score is the harmonic mean of both precision and recall, providing a balanced indication of the model's ability to predict the positive class. There is typically a tradeoff made between precision and recall – models with high recall are able to identify a higher proportion of the positive class, often through less stringency with assigning the positive label. This comes at a cost of more false positives, which hampers the model's precision. This tradeoff is particularly apparent for the SVM-GAK model, which has the highest recall of 0.70 – it is able to identify 70 % of all actual *go*'s, but only 55 % of the events for which it predicts *go*'*s* are actual *go* events.

It is observed that the best overall model is LSTM, which has the highest accuracy, precision, and F1-score. This does come with a cost on recall as is expected due to the aforementioned precision-recall tradeoff. Since the purpose of this work is to assess the ability of AI to match operator-made decisions, only the F1-score, which is a general indicator of model performance for imbalanced classes, is the prioritized metric. The F1-score does not value either precision or recall over the other – operationally, where having a higher recall is desirable (to not miss any *go*'*s*, even at the cost of false positives), using an F2-score, which prioritizes recall, or directly using recall might be more relevant.

4.2 Feature Importance

To better interpret the model outcomes, it can be beneficial to understand which features impact the model's performance the most. Since the LSTM model is the best performing model (based on the F1-score) of the ones tested, feature importance is applied to the LSTM model by masking the values of one feature column at a time and running it on a test set. This effectively hides the values of this feature and associated weights, so the model is unable to use insights from it when classifying unseen data. Figure 4 summarizes the effect of removing features on the F1-score for a sample LSTM run with an F1-score of 0.76.

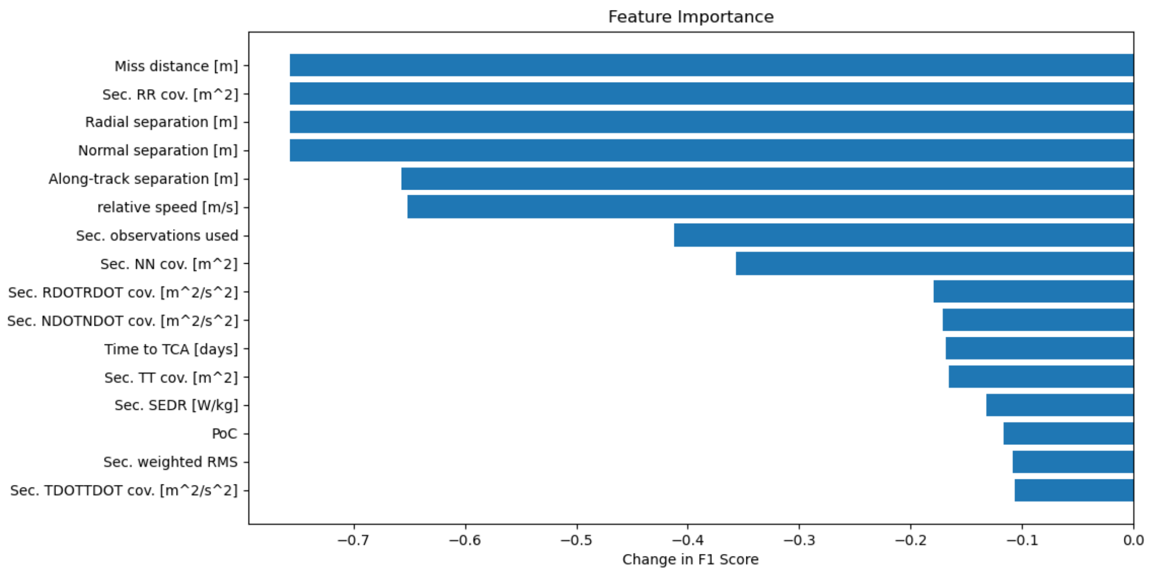


Fig. 4. Features (defined in Section 3.2) ranked according to their contributions to model performance.

The features found to impact model performance the most are miss distance, radial position uncertainties of the secondary object, and the relative position between the two objects. These are in alignment with features prioritized by analysts as well. However, arguably the most important feature for decision-making, PoC, is weighted low by the LSTM model. The model is able to make roughly the same quality of predictions even if it does not consider PoC, as long as it has the other features. This can occur if there are many events for which the PoC trend of both *go* and *no go* events is fairly similar, compared to that of other features. For instance, this applies to cases where an event displaying a critical PoC trend is still characterized as a *no go* due to large uncertainties. Thus, AI models do not necessarily weight specific features the same way that decision-making analysts do. However, much like an analyst considers the full context of information provided across multiple features – especially for events lying close to the PoC threshold or with irregular trends – the LSTM model attempts to do so as well, likely making it an ideal candidate for further applications to this problem.

5 CONCLUSIONS

This work sets out to investigate the practicability of using AI to automate the *go*/*no go* decision-making process by treating it as a time series classification problem. 200 events with a probability of collision (PoC) around the typical maneuver threshold of 1E-4 are classified by analysts at the German Space Operations Center (GSOC). 4 AI models – K-Nearest Neighbors with Dynamic Time Warping (KNN-DTW), KNN with Symbolic Aggregate Approximation (KNN-SAX), Support Vector Machine with the Global Alignment Kernel (SVM-GAK), and Long Short-Term Memory (LSTM) – are evaluated based on their ability to match the decisions made by the GSOC analysts. The LSTM model yields the best accuracy (72 %), precision (69 %), and F1-score (65 %), and is thus worth investigating further for AI applications towards decision-making in this context. SVM-GAK yields the highest recall of 70 %, making it a promising choice for minimizing false negatives. Feature importance screening on the LSTM model pointed to position-related features (miss distance, radial position covariance, and relative position) as being the most influential on the model's performance. All in all, there is potential for AI to contribute to decision-making in this context. Continued research into different types of models, feature selection and engineering, and hyperparameter optimization is essential for the development of a truly reliable and robust AI-based decision-making system.

6 ACKNOWLEDGEMENTS

This work is funded by the Munich Aerospace scholarship program.

7 REFERENCES

1. LeoLabs. What's up in LEO? Insights and analysis from 2022. https://leolabs.space/article/leo-annual-review-2022/, 2023.
2. ESA Space Debris Office. *ESA's Annual Space Environment Report*, 2023.
3. D. McKnight, E. Dale, R. Bhatia, M. Patel, C. Kunstadter. A Map of the Statistical Collision Risk in LEO. In *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*, 2022.
4. United States Government Accountability Office. *Large Constellations of Satellites*, 2022.
5. T. Uriot, D. Izzo, L. Simões, R. Abay, N. Einecke, S. Rebhan, J. Martinez-Heras, F. Letizia, J. Siminski, K. Merz. Spacecraft collision Avoidance challenge: Design and results of a machine learning competition. *Astrodynamics* **6**, 121–140, 2022.
6. R. Abay, F. Caldas, M. Filipe, M. Guimarães. Benchmarking Machine Learning Models for Collision Risk Prediction in Low Earth Orbit. In *Proceedings of the 8th European Conference on Space Debris*, 2021.
7. V. Schaus, T. Andriof, C. Borrett, I. Burmeister, F. Cabral, et al. First Results of ESA's Collision Risk Estimation and Automated Mitigation (CREAM) Programme. In *Proceedings of the 73rd International Astronautical Congress (IAC)*, 2022.
8. F. Pinto, G. Acciarini, S. Metz, S. Boufelja, S. Kaczmarek, K. Merz, et al. Towards Automated Satellite Conjunction Management with Bayesian Deep Learning. *AI for Earth Sciences Workshop, NeurIPS*, 2020.
9. I. F. Stroe, A.D. Stanculescu, P. B. Ilioaica, C. F. Blaj, et al. AUTOCA Autonomous Collision Avoidance System. In *Proceedings of the 8th European Conference on Space Debris*, 2021.
10. C. P. Hernández, D. Lubián-Arenillas, C. P. Periañez, J. T. Velez, A. Solomon. Should I stay or should I go? Machine Learning applied to Conjunction Analysis. In *Proceedings of the 73rd International Astronautical Congress (IAC)*, 2022.
11. L. Sánchez, M. Vasile, E. Minisci. On the use of machine learning and evidence theory to improve collision risk management. *2nd IAA Conference on Space Situational Awareness (ICSSA)*, 2020.
12. S. Aida, M. Kirschner. Operational Results of Conjunction Assessment and Mitigation at German Space Operations Center. *Transactions of the Japanese Society for Aeronautical and Space Sciences* **28**, 23-28, 2012.
13. J. Faouzi, Time Series Classification: A review of Algorithms and Implementations. *Machine Learning*, 2022.
14. A. Bier, A. Jastrzębska, P. Olszewski. Variable-Length Multivariate Time Series Classification Using ROCKET: A Case Study of Incident Detection. *IEEE Access* **10**, 2022.
15. R. Tavenard, et al. Tslearn, A Machine Learning Toolkit for Time Series Data. *Journal of ML Research* **21**, 1-6, 2020.
16. F. Chollet et al. Keras. https://github.com/keras-team/keras, 2015.
17. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, 2016.
18. H. Sakoe, S. Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **26**, 1978.
19. A. Bagnall, J. Lines, A. Bostrom, J. Large, E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery* **31**, 606-660, 2017.
20. J. Lin, E. Keogh, L. Wei, S. Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery* **15**, 107-144, 2007.
21. M. Cuturi, J-P. Vert, et al. A Kernel for Time Series Based on Global Alignments. *IEEE International Conference*, 2007.
22. L. Chen, X. Bai, Y. Liang, K. Li, *Orbital Data Applications for Space Objects*, 2017.
23. M. Sokolova, G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management* **45**, 427-437, 2009.
24. S. Hochreiter, J. Schmidhuber. Long Short-Term Memory. *Neural Computation* **9**(8), 1735-1780, 1997.